An $O^*(3.52^{3k})$ Parameterized Algorithm for 3-Set Packing*

Jianxin Wang and Qilong Feng

School of Information Science and Engineering, Central South University
jxwang@mail.csu.edu.cn

Abstract. Packing problems have formed an important class of NP-hard problems. In this paper, we provide further theoretical study on the structure of the problems, and design improved algorithm for packing problems. For the 3-Set Packing problem, we present a deterministic algorithm of time $O^*(3.52^{3k})$, which significantly improves the previous best algorithm of time $O^*(4.61^{3k})$.

1 Introduction

In the complexity theory, packing problem forms an important class of NP-hard problems, which are used widely in scheduling and code optimization fields. We first give some related definitions [1].

Assume all the elements used in this paper are from U.

Set Packing: Given a pair (S, k), where S is a collection of n sets and k is an integer, find a largest subset S' such that no two sets in S' have the common elements.

(Parameterized) 3-Set Packing: Given a pair (S, k), where S is a collection of n sets and k is an integer, each set contains 3elements, either construct a k-packing or report that no such packing exists.

3-Set Packing Augmentation: Given a pair (S, P_k), where S is a collection of n sets and P_k is a k-packing in S, either construct $(k+1)$-packing or report that no such packing exists.

Recently, Downey and Fellows [2] proved that the 3-D Matching problem is Fixed Parameter Tractable (FPT), and gave an algorithm with time complexity $O^*((3k)!(3k)^{9k+1})$, which can be applied to solve 3-Set Packing problem. Jia, Zhang and Chen [3] reduced the time complexity to $O^*((5.7k)^k)$ using greedy localization method. Koutis [4] proposed a randomized algorithm with time

* This work is in part supported by the National Natural Science Foundation of China under Grant No. 60773111 and No. 60433020, Provincial Natural Science Foundation of Hunan (06JJ10009), the Program for New Century Excellent Talents in University No. NCET-05-0683 and the Program for Changjiang Scholars and Innovative Research Team in University No. IRT0661.

M. Agrawal et al. (Eds.): TAMC 2008, LNCS 4978, pp. 82–93, 2008.

complexity $O^*(10.88^{3k})$ and a deterministic algorithm with time complexity at least $O^*(32000^{3k})$. Fellows et al [5] gave an algorithm with time complexity at least $O^*(12.67^{3k}T(k))$ for the 3-D Matching problem, (based on current technology, $T(k)$ is at least $O^*(10.4^{3k})$). Kneis, Moelle, S.R and Rossmanith [6] presented a deterministic algorithm of time $O^*(16^{3k})$ using randomized divide and conquer. Chen, Lu, S.H.S and Zhang [7] gave a randomized algorithm with time complexity $O^*(2.52^{3k})$ based on the divide and conquer method, whose deterministic algorithm is of time complexity $O^*(12.8^{3k})$. Liu, Lu, Chen and H Sze [1] gave a deterministic algorithm of time $O^*(4.61^{3k})$ based on greedy localization and color coding, which is currently the best result in the world.

In this paper, we will discuss how to construct a $(k+1)$-packing from a k-packing, so as to solve the 3-Set Packing problem. After further analyzing the structure of the problem, we can get the following property: if the 3-Set Packing Augmentation problem can not be solved in polynomial time, then each set in P_{k+1} should contain at least one element of P_k. Based on the above property, we can get a randomized algorithm of time $O^*(3.52^{3k})$ using randomized divide and conquer. According to the structure analysis and the derandomization method given in [8], we can get a deterministic algorithm with the same time complexity $O^*(3.52^{3k})$, which greatly improves the current best result $O^*(4.61^{3k})$.

2 Related Terminology and Lemmas

We first introduce two important lemmas [1].

Lemma 1. *For any constant $c > 1$, the 3-Set Packing Augmentation problem can be solved in time $O^*(c^k)$ if and only if the 3-Set Packing problem can be solved in time $O^*(c^k)$.*

Lemma 2. *Let (S, P_k) be an instance of 3-Set Packing Augmentation, where P_k is a k-packing in S. If S also has $(k+1)$-packings, then there exists a $(k+1)$-packing P_{k+1} in S such that every set in P_k contains at least two elements in P_{k+1}.*

By lemma 1, reducing the time complexity of 3-Set Packing Augmentation problem is our objective.

For the convenience of analyzing the structure of the 3-Set Packing Augmentation problem, we give the following definitions.

Definition 1. Let (S, P_k) be an instance of 3-Set Packing Augmentation problem, where P_k is a k-packing in S. Assume there exists P_{k+1}, for a certain $(k+1)$-packing P and a set ρ_i in P_k, if only one element of ρ_i is contained in P, ρ_i is called *1-Set*; if no element of ρ_i is contained in P, it is called *0-Set*. The collection of all the 1-Set and 0-Set in P_k is called $(1,0)$-*Collection* of the $(k+1)$-packing P. If the $(1,0)$-Collection of P is null, then P is called a $(0,1)$-*free packing*.

We need to point out that: different $(k+1)$-packing may have different $(1,0)$-Collection.

It is easy to get the following lemma from relation between P_k and P_{k+1}.

Lemma 3. *Let* (S, P_k) *be an instance of* 3*-Set Packing Augmentation problem, where* P_k *is* k*-packing in* S*. Assume there exists* P_{k+1}*, any* $(k+1)$*-packing can be transformed into a* $(0,1)$*-free packing in polynomial time.*

Proof. For an instance of 3-Set Packing Augmentation problem (S, P_k). Assume there exists P_{k+1}, for any $(k+1)$-packing P, do the following process.

Find out all the 1-Set and 0-Set in P_k, denoted by W. For each set ρ_i in W, discuss it in the following two cases.

Case 1: ρ_i is a 1-Set. Assume one element a in ρ_i is contained in P and the set ρ'_j in P contains the element a. Use ρ_i to replace ρ'_j in P such that the number of 1-Set in P_k is reduced by one. Because of the replacement, there may produce new 1-Set or 0-Set in P_k. Find out all the new 1-Set and 0-Set in P_k. If a new 1-Set or 0-Set has not existed in W, add this set into W.

Case 2: ρ_i is a 0-Set. Since P_k has k sets and P_{k+1} has $k+1$ sets, there must exists one set ρ'_j in P that is not contained in P_k. Use ρ_i to replace ρ'_j in P such that the number of 0-Set in P_k is reduced by one. Because of the replacement, there may produce new 1-Set or 0-Set in P_k. Find out all the new 1-Set and 0-Set in P_k. If a new 1-Set or 0-Set has not existed in W, add this set into W.

After processing all the sets in W, there are no 1-Set and 0-Set in P_k. Therefore, the $(k+1)$-packing P is converted into a $(0,1)$-free packing.

Now, we prove that the above process can be done in polynomial time.

In order to find out all the 1-Set and 0-Set, for each set ρ_i in P_k, we need to consider all the sets in P. Obviously, the time complexity of this process is bounded by $O(k^2)$. When a set in P is replaced by 1-Set or 0-Set,it needs to redetermine the 1-Set and 0-Set in P_k. The whole time complexity of this process is bounded by $O(k^3)$. This completes the proof of the lemma. □

Based on the lemma 3, we can get following lemma.

Lemma 4. *Let* (S, P_k) *be an instance of* 3*-Set Packing Augmentation problem, where* P_k *is a* k*-packing in* S*. Assume there exists* P_{k+1}*, for any* $(k+1)$*-packing* P*, if* P *is a* $(0,1)$*-free packing, then each set in* P_k *should have at least 2 elements be contained in* P*.*

Proof. Assume there exists P_{k+1} in S, for any $(k+1)$-packing P, if P is a $(0,1)$-free packing, the $(1,0)$-Collection of P is null, that is, there is no 1-Set or 0-Set in P_k. Therefore, each set in P_k should have at least 2 elements be contained in P. □

Combing lemma 4 with the structure analysis of P_{k+1}, we can get the following lemma.

Lemma 5. *Given an instance of* 3*-Set Packing Augmentation problem* (S, P_k)*, where* P_k *is* k*-packing in* S*. For any* $(0,1)$*-free packing* P*, assume there are* $2k+x$ $(0 \le x \le k)$ *elements from* P_k *contained in* P*, if the* 3*-Set Packing Augmentation problem can not be solved in polynomial time, each set in* P *should contain at least one of those* $2k+x$ *elements.*

Proof. By the lemma 4, each $(0,1)$-free packing contains at least $2k$ elements of P_k.

We use contradiction method to prove. Assume that: although the 3-Set Packing Augmentation problem can not be solved in polynomial time, one or more sets in P contain none of those $2k+x$ elements.

Assume that there is a set α in P containing none of those $2k+x$ elements. Except those $2k+x$ elements, other elements in P_k are definitely not in P. Therefore, α and all sets in P_k have no common elements.

According to the relation between elements and sets, construct the bipartite graph $G=(V_1 \cup V_2, E)$, where the vertices in V_1 correspond to the elements in U, and the vertices in V_2 denote the sets in S. If an element is contained in a set, then the corresponding vertices will be connected by an edge. In graph G, we can find out all the sets having no common elements with P_k, thus, α is definitely in those sets. A $(k+1)$-packing can be constructed by the k sets in P_k and α, which can be done in polynomial time. This contradicts with the assumption. This completes the proof of the lemma. □

3 The Randomized Algorithm

In this part, randomized divide and conquer will be used efficiently to solve the 3-Set Packing Augmentation problem. Based on the lemma 3, we can assume that all the $(k+1)$-packing used in the following are $(0,1)$-free packing.

By lemma 5, we can get the following lemma.

Lemma 6. *Given an instance of* 3*-Set Packing Augmentation problem* (S, P_k)*, where* P_k *is a* k*-packing in* S*. Assume there exists* P_{k+1}*, for a* $(k+1)$*-packing* P*, if* P *can not be found in polynomial time,* P *is composed of the following there parts.*

(1) *P has r sets, each of which contains only one element of P_k, $0 \le r \le \lceil \frac{k+3}{2} \rceil$.*

(2) *P has s sets, each of which contains only two elements of P_k, $0 \le s \le k+1$.*

(3) *P has t sets, each of which contains three elements of P_k, $0 \le t \le k-1$.*

where $r+s+t=k+1$.

Proof. By lemma 5, if there exists P_{k+1} and could not find a $(k+1)$-packing in polynomial time, each set in P_{k+1} should contains at least one element of P_k. Therefore, for the $(k+1)$-packing P, each set in P may contain 1, 2 or 3 elements of P_k, which is one of the three types given in the lemma.

Now we will prove that there are at most $\lceil \frac{k+3}{2} \rceil$ sets in P, each of which contains only one element of P_k. Assume that P contains $2k+x$ $(0 \le x \le k)$ elements of P_k. If $s=0$ and each set in P has already contained one of those $2k+x$ elements. When the remaining $2k+x-(k+1)=k+x-1$ elements are used to form the sets containing three elements of P_k, r gets the maximum value: $k+1-\lfloor \frac{k+x-1}{2} \rfloor \le \lceil \frac{k+3}{2} \rceil$.

When each set in P contains only two elements of P_k, s gets the maximum value $k+1$, thus, $s \le k+1$.

Because each set in P contains at least one element of P_k, the maximum number of sets in P containing three elements of P_k is $k-1$, thus, $t \le k-1$. □

Let C_i $(1 \le i \le 3)$ denote all the sets having i common elements with P_k, which can be found in polynomial time based on the relation of elements and sets. Assume U_{P_k} denotes the $3k$ elements in P_k and U_{S-P_k} denotes the elements in $S - P_k$. Therefore, each set in C_2 contains 2 elements from U_{P_k}, and each set in C_3 contains 3 elements from U_{P_k}. Let $U_{C_2-P_k}$ denote the elements in C_2 but not in U_{P_k}, then $U_{C_2-P_k} \subseteq U_{S-P_k}$.

By lemma 6, if P_{k+1} exists, there are r sets in P_{k+1} such that each of which contains only one element of P_k, which are obviously included in C_1. To find the r elements from U_{P_k}, there are $\binom{3k}{r}$ enumerations. Let H be the collection of sets in C_1 containing one of those r elements.

Assume $U_{P_{k+1}-P_k}$ denotes all the elements in P_{k+1} but not in P_k, and the size of the $U_{P_{k+1}-P_k}$ is denoted by $y = |U_{P_{k+1}-P_k}|$. By lemma 5, P_{k+1} contains at least $2k$ elements of U_{P_k}, thus, $U_{P_{k+1}-P_k}$ contains at most $k+3$ elements of U_{S-P_k}, that is, $y \le k+3$. It can be seen that the elements in $U_{P_{k+1}-P_k}$ are either in H or in $C_2 \cup C_3$. Assume that $U'_{P_{k+1}-P_k}$ denotes the elements of $U_{P_{k+1}-P_k}$ belonging to H. When the elements in $U_{P_{k+1}-P_k}$ are partitioned, the probability that the elements in $U'_{P_{k+1}-P_k}$ are exactly partitioned into H is $\frac{1}{2^y}$.

The general ideal of our randomized algorithm is as follows:

Divide P_{k+1} into two parts to handle, one of which is in H and the other in $C_2 \cup C_3$. For the part contained in $C_2 \cup C_3$, we use dynamic programming to find a $(k+1-r)$-packing; For the part in H, we use randomized divide and conquer to handle.

3.1 Use Dynamic Programming to find a $(k+1-r)$-Packing in $C_2 \cup C_3$

For the convenience of describing the algorithm, we first give the concept of symbol pair. For each set $\rho_i \in C_2$, the elements from U_{P_k} in set ρ_i is called a symbol pair.

The algorithm of finding a $k' = k+1-r$ $(k' \le k+1)$ packing in $C_2 \cup C_3$ is given in figure 1.

Theorem 1. *If there exists k'-packing in $C_2 \cup C_3$, algorithm SP will definitely return a collection of symbol pairs and 3-sets with size k', and the time complexity is bounded by $O^*(2^{3k})$.*

Proof. If there exists k'-packing in $C_2 \cup C_3$, assume that the number of sets in C_2 contained in the k'-packing is k'', $0 \le k'' \le k'$. Therefore, the k'' sets of k'-packing in C_2 can form a k''-packing. We need to prove the following two parts.

(1) After the execution of the for-loop in step 3, Q_1 must contain a collection of symbol pairs with size k''.

(2) After the execution of the for-loop in step 5, Q_1 must contain a collection of symbol pairs and 3-sets with size k'.

The proof of the first part is as follows.

It can be seen from step 3.1-3.4 that the C' added into Q_1 in the step 3.7 is a collection of symbol pairs from the right packing. We get a induction for the i in

Algorithm SP
Input: C_2, C_3, k', U_{P_k}
Output: if there exists k'-packing in $C_2 \cup C_3$, return a collection of symbol pairs and 3-sets with size k'
1. assume the elements in $U_{C_2-P_k}$ are $x_1, x_2, \ldots x_m$;
2. Q_1=$\{\phi\}$; Q_{new}=$\{\phi\}$;
3. **for** $i = 1$ **to** m **do**
3.1 **for** each collection C in Q_1 **do**
3.2 **for** each 3-set ρ in C_2 having element x_i **do**
3.3 **if** C has no common element with ρ **then**
3.4 $C' = C \cup \{$elements in ρ belonging to $U_{P_k}\}$;
3.5 **if** C' is not larger than k' and no collection in Q_{new} has used exactly the same elements as that used in C' **then**
3.6 add C' into Q_{new};
3.7 $Q_1 = Q_{new}$;
4. assume the 3-sets in C_3 are $z_1, z_2, \ldots z_l$;
5. **for** $h = 1$ **to** l **do**
5.1 **for** each collection C in Q_1 **do**
5.2 **if** C does not have common elements with z_h **then**$C' = C \cup \{z_h\}$;
5.4 **if** C' is not larger than k' and no collection in Q_{new} has used exactly the same elements as that used in C' **then** add C' into Q_{new};
5.6 $Q_1 = Q_{new}$;
6. **if** there is a collection of symbol pairs and 3-sets with size k' in Q_1 **then** return the collection.

Fig. 1. Use dynamic programming to find a $(k + 1 - r)$-packing in $C_2 \cup C_3$

the step 3 so as to prove that: if there exists k''-packing in C_2, Q_1 must contain a collection of symbol pairs with size k''.

There are m different elements in $U_{C_2-P_k}$: $x_1, x_2, \ldots x_m$. For any arbitrary i $(1 \le i \le m)$, assume that X_i denotes all the sets containing the element in $\{x_1, x_2, \ldots x_i\}$. Therefore, we only need to prove the following claim.

Claim 1. If there exists a j-packing symbol pairs collection P_j in X_i, then after i-thexecution of the for-loop in step 3, Q_1 contains a j-packing symbol pairs collection P'_j, which uses the same $2j$ elements with P_j.

In the step 2, Q_1=$\{\phi\}$. Thus, if X_i has a 0-packing, the claim is true.

When $i \ge 1$, assume there exists a j-packing symbol pair collection $P_j = \{\varphi_{l_1}, \varphi_{l_2}, \ldots, \varphi_{l_j}\}$, where $1 \le l_1 < l_2 < \cdots < l_j \le i$, then there must exists a $(j-1)$-packing symbol pair collection $P_{j-1} = \{\varphi_{l_1}, \varphi_{l_2}, \ldots, \varphi_{l_j-1}\}$ in X_{l_j-1}. By the induction assumption, after the $(l_j - 1)$-th execution of for-loop in step 3, Q_1 contains a $(j-1)$-packing symbol pairs collection P'_{j-1}, which use the same $2(j-1)$ elements of U_{P_k} with P_{j-1}. By the assumption, X_i contains a j-packing symbol pair collection P_j. Therefore, when the set containing the φ_{l_j} is considered in step 3.2, the elements belonging to U_{P_k} in P'_{j-1} are totally different from the elements in φ_{l_j}. As a result, if there is no collection of symbol pairs in Q_1 containing the same $2j$ elements with $P'_{j-1} \cup \{\varphi_{l_j}\}$, the j-packing symbol pairs $P'_{j-1} \cup \{\varphi_{l_j}\}$ will be added into Q_1. Because all the collection of symbol pairs in Q_1 are not removed from Q_1 and $l_j \leq i$, after the i-th execution of the for-loop in step 3, Q_1 must contains a j-packing symbol pairs collection using the same $2j$ elements with P_j. When $i = m$, if there exists k''-packing $P_{k''}$ in C_2, Q_1 must contain a collection of symbol pairs using the same $2k''$ elements of U_{P_k} with $P_{k''}$.

The proof of the second part is similar to the first part, which is neglected here.

At last, we analyze the time complexity of algorithm SP. If considering C_2 only, for each j $(0 \leq j \leq k')$ and any subset of U_{P_k} containing $2j$ elements, Q_1 record at most one collection symbol pairs using those $2j$ elements, thus, Q_1 contains at most $\sum_{j=0}^{k'+1} \binom{3k}{2j}$ collections. If considering C_3 only, for each j $(0 \leq j \leq k')$ and any subset of containing $3j$ elements, Q_1 record at most one collection of 3-sets using those $3j$ elements, thus, Q_1 contains at most $\sum_{j=0}^{k'-1} \binom{3k}{3j}$ collections. Therefore, the time complexity of algorithm SP is bounded by $\max\{O^*(\sum_{j=0}^{k'+1} \binom{3k}{2j}), O^*(\sum_{j=0}^{k'-1} \binom{3k}{3j})\} = O^*(2^{3k})$. □

If there exists k'-packing in $C_2 \cup C_3$, the collection returned by algorithm SP may contain symbol pairs, which can be converted into 3-sets using the bipartite maximum matching.

3.2 Use Randomized Divide and Conquer to find a r-Packing in H

Assume that we have already picked r $(0 \leq r \leq \lceil \frac{k+3}{2} \rceil)$ elements from U_{P_k}, and let H be the collection of sets in C_1 containing one of those r elements. The algorithm of finding a r-packing in H is given in figure 2.

Theorem 2. *If H has r-packing, algorithm RSP will return a collection D containing the r-packing with probability larger than 0.75, and the time complexity is bounded by $O^*(4^{2r})$.*

Proof. Algorithm RSP divides H into two parts to handle: H_1, H_2. If there exists r-packing P_r, P_r has $2r$ elements of U_{S-P_k}, denoted by $U_{P_r-P_k}$. Assume that $U'_{P_r-P_k}$ denotes the elements belonging to $U_{P_r-P_k}$ in H_1, thus, the elements belonging to $U_{P_r-P_k}$ in H_2 can be denoted by $U_{P_r-P_k} - U'_{P_r-P_k}$. Mark all the elements from U_{S-P_k} in H_1 and H_2 with red and blue colors. When elements in $U'_{P_r-P_k}$ are exactly marked with red and elements in $U_{P_r-P_k} - U'_{P_r-P_k}$ are marked with blue, it is called that elements in H_1 and H_2 are rightly marked, which occurs with probability $\frac{1}{2^{2r}}$. Therefore, the probability that the elements in H_1 and H_2 are not rightly marked is $1 - \frac{1}{2^{2r}}$.

If there exists r-packing in H, let δ_r be the probability that algorithm RSP can not find the r-packing. In the step 3, H is divided into two parts: H_1, H_2. Therefore, the probability that RSP(H_1) and RSP(H_2) can not find the corresponding packing respectively is $\delta_{r/2}$. After $3 \cdot 2^{2r}$ iterations, the probability that algorithm RSP could not find the P_{k+1} is $(1 - \frac{1}{2^{2r}} + \frac{1}{2^{2r-1}} \cdot \delta_{r/2})^{3 \cdot 2^{2r}}$. We need

Algorithm RSP
Input: H, r
Output: return a collection D of packings

1. **if** $r = 0$ **then** return ϕ;
2. **if** $r = 1$ **then** return H;
3. randomly pick $\lceil \frac{r}{2} \rceil$ elements from the r elements, and let H_1 denote all the sets containing one of those $\lceil \frac{r}{2} \rceil$ elements;
4. $H_2 = H - H_1$; $D = \phi$;
5. **for** $3 \cdot 2^{2r}$ times **do**

5.1 mark all the elements from U_{S-P_k} in H_1 and H_2 with red and blue;
for each set ρ in H_1, if the colors of the elements belonging to U_{S-P_k} are not both red, delete the set ρ;
for each set ρ' in H_2, if the colors of the elements belonging to U_{S-P_k} are not both blue, delete the set ρ';

5.2 D_1=RSP(H_1, $\lceil \frac{r}{2} \rceil$);

5.3 D_2=RSP(H_2, $\lfloor \frac{r}{2} \rfloor$);

5.4 **for** each packing α in D_1 **do**
  for each packing β in D_2 **do**
    if there does not exist $\alpha \cup \beta$ in D, add $\alpha \cup \beta$ into D;

6. return D;

Fig. 2. Use randomized divide and conquer to find a r-packing in H

to prove that: for any r, $\delta_r \leq 1/4$. It is obvious that $\delta_1 = 0$. Assume $\delta_{r/2} \leq 1/4$. By the induction assumption, we can get that: $\delta_r = (1 - \frac{1}{2^{2r}} + \frac{1}{2^{2r-1}} \cdot \delta_{r/2})^{3 \cdot 2^{2r}} \leq (1 - \frac{1}{2^{2r}} + \frac{1}{2^{2r-1}} \cdot 1/4)^{3 \cdot 2^{2r}} = (1 - \frac{1}{2^{2r+1}})^{\frac{3}{2} \cdot 2^{2r+1}} \leq e^{-3/2} < 1/4$.

Let T_r denote the number of recursive calls in algorithm RSP, then $T_r \leq 3 \cdot 2^{2r} \cdot (T_{\lceil \frac{r}{2} \rceil} + T_{\lfloor \frac{r}{2} \rfloor}) \leq 3 \cdot 2^{2r+1} \cdot T_{\lceil \frac{r}{2} \rceil} = O(3^{\log 2r} 4^{2r}) = O((2r)^{\log 3} 4^{2r}) = O^*(4^{2r})$. □

3.3 The General Algorithm for 3-Set Packing Augmentation

Based on the above two algorithm, the general algorithm for 3-Set Packing Augmentation problem is given in figure 3.

Theorem 3. *If S has $(k+1)$-packing, algorithm GSP will return the $(k+1)$-packing with probability larger than 0.75, and the time complexity is bounded by $O^*(3.52^{3k})$.*

Proof. In the above algorithm, we need to consider all the enumeration of r. If S has a $(k+1)$-packing, there must exist a r and an enumeration satisfying the condition. The algorithm divides P_{k+1} into two parts to handle, one of which is in H and the other in $C_2 \cup C_3$. When elements in $U'_{P_r - P_k}$ are exactly marked with white and elements in $U_{P_r - P_k} - U'_{P_r - P_k}$ are marked with black, it is called that elements in U_{S-P_k} are rightly marked, which occurs with probability $\frac{1}{2^y}$.

Algorithm GSP
Input: S, k
Output: whether there is a $(k+1)$-packing in S

1. **for** $r = 0$ **to** $\lceil \frac{k+3}{2} \rceil$ **do**
1.1 enumerate r elements from U_{P_k}, and get $\binom{3k}{r}$ enumerations;
1.2 **for** each enumeration **do**
let H be the collection of sets in C_1 containing one of those r elements;
1.3 **for** $24 \cdot 2^k$ times **do**
$C_2' = C_2$; $C_3' = C_3$;
for each element a in U_{P_k}, if a belongs to H, then delete all the sets in $C_2' \cup C_3'$ containing a;
use colors black and white to mark all the elements in U_{S-P_k};
for each set ρ in C_2', if the color of the element belonging to U_{S-P_k} is not black, delete the set ρ;
for each set ρ' in H, if the colors of the elements belonging to U_{S-P_k} are not both white, delete the set ρ';
$Q_2 =$SP$(C_2', C_3', k+1-r, U_{P_k})$;
$Q_3 =$RSP(H, r);
use the bipartite maximum matching algorithm to convert the symbol pairs in Q_2 into 3-sets;
if Q_2 is a $(k+r-1)$-packing and Q_3 has a r-packing
then return the $(k+1)$-packing; stop;
2. return no $(k+1)$-packing in S;

Fig. 3. The general algorithm for 3-Set Packing Augmentation

Therefore, the probability that the elements in U_{S-P_k} are not rightly marked is $1 - \frac{1}{2^y}$. If S has P_{k+1}, let δ_k denote the probability that algorithm GSP can not find the P_{k+1}. If H has r-packing, let δ_r denote the probability that algorithm RSP can not find the r-packing. By theorem 3, we know that $\delta_r \le 1/4$. If P_{k+1} exists, for a certain r and an enumeration, after $24 \cdot 2^k$ iterations of step 1.3, the probability that algorithm GSP could not find the P_{k+1} is $(1 - \frac{1}{2^y} + \frac{1}{2^{y-1}} \cdot \delta_r)^{24 \cdot 2^k}$, that is,

$\delta_k = (1 - \frac{1}{2^y} + \frac{1}{2^{y-1}} \cdot \delta_r)^{24 \cdot 2^k} \le (1 - \frac{1}{2^y} + \frac{1}{2^{y-1}} \cdot \frac{1}{4})^{24 \cdot 2^k} = (1 - \frac{1}{2^{y+1}})^{24 \cdot 2^k} \le e^{-3/2} < 1/4$.

By theorem 1, If there exists $(k+1-r)$-packing in $C_2 \cup C_3$, algorithm SP will definitely return a collection of symbol pairs and 3-sets with size $k+1-r$. By theorem 2, if H has r-packing, algorithm RSP will return the r-packing with probability larger than 0.75. Therefore, if S has $(k+1)$-packing, algorithm GSP will return the $(k+1)$-packing with probability larger than 0.75.

Now we analyze time complexity of the above algorithm. For each r, there are $\binom{3k}{r}$ ways to enumerate r elements from U_{P_k}. By theorem 1, the time complexity of algorithm SP is bounded by $O^*(2^{3k})$. By theorem 2, the time complexity of algorithm RSP is $O^*(4^{2r})$. Because of $0 \leq r \leq \lceil \frac{k+3}{2} \rceil$, the running time of algorithm RSP is bounded by $O^*(4^k)$. Using bipartite maximum matching algorithm to covert symbol pairs in Q_2 can be done in polynomial time. Therefore, the total time complexity of algorithm GSP is $\sum_{r=0}^{\lceil \frac{k+3}{2} \rceil} \binom{3k}{r}(2^k(2^{3k-r}+4^k)) = O^*(3.52^{3k})$. $\square$

4 Derandomization

When there exists $(k+1)$-packing, in order to make failure impossible, we need to derandomize the above algorithm. We first point out that: partitioning a set means dividing the set into two parts.

Because the size of the $U_{P_{k+1}-P_k}$ is y, there are 2^y ways to partition $U_{P_{k+1}-P_k}$. Therefore, there must exist a partition satisfying the following property: elements in $U'_{P_{k+1}-P_k}$ are exactly partitioned into H, and elements in $U_{P_{k+1}-P_k} - U'_{P_{k+1}-P_k}$ are exactly in $C_2 \cup C_3$. However, the problem is that: $U_{P_{k+1}-P_k}$ is unknown.

Naor, Schulman and Srinivasan [9] gave the solution for the above problem. Moreover, Chen and Lu [8] presented a more detailed description of that method.

Now we introduce a very important lemma in [10].

Lemma 7. *Let n, k be two integers such that $0 < k \leq n$. There is an (n,k)-universal set P of size bounded by $O(n2^{k+12\log^2 k+12\log k+6})$, which can be constructed in time $O(n2^{k+12\log^2 k+12\log k+6})$.*

The (n,k)-universal set in the above lemma is a set F of splitting functions, such that for every k-subset W of $\{0,1,\cdots,n-1\}$ and any partition (W_1,W_2) of W, there is a splitting function f in F that implements (W_1,W_2).

In the construction of the above lemma, Chen and Lu [8] constructed a fuction $h(x)$=$((ix \bmod p) \bmod k^2)$ from $\{0,1,\cdots,n-1\}$ to $\{0,1,\cdots,k^2-1\}$, and used the fact that there are at most $2n$ $h(x)$ to get the above lemma. However, the bound $2n$ is not tight. Now, we introduce an important lemma in [9].

Lemma 8. *There is an explicit (n,k,k^2)-splitter $A(n,k)$ of size $O(k^6 \log k \log n)$.*

In the above lemma, the (n,k,k^2)-splitter $A(n,k)$ denotes the function from $\{0,1,\cdots,n-1\}$ to $\{0,1,\cdots,k^2-1\}$. Thus, the number of functions from $\{0,1,\cdots,n-1\}$ to $\{0,1,\cdots,k^2-1\}$ are bounded by $O(k^6 \log k \log n)$.

Based on lemma 7, lemma 8, we can get the following lemma.

Lemma 9. *Let n, k be two integers such that $0 < k \leq n$. There is an (n,k)-universal set P of size bounded by $O(\log n 2^{k+12\log^2 k+18\log k})$, which can be constructed in time $O(\log n 2^{k+12\log^2 k+18\log k})$.*

By the lemma 9, we can get the following theorem.

Theorem 4. *3-Set Packing Augmentation problem can be solved deterministically in time $O^*(3.52^{3k})$.*

Proof. Given an instance of 3-Set Packing Augmentation problem (S, P_k), if there exists P_{k+1}, by lemma 5, P_{k+1} contains at least $2k$ elements of U_{P_k}, thus, $U_{P_{k+1}-P_k}$ contains at most $k+3$ elements of U_{S-P_k}. After picking r elements from U_{P_k}, P_{k+1} is divided into two parts to handle in the randomized algorithm, one of which is in H and the other in $C_2 \cup C_3$. By lemma 9, we can construct the $(S-P_k, k+3)$-universal set, whose size is bounded by $O(\log n 2^{k+3+12\log^2(k+3)+18\log(k+3)})$.

For each partition to U_{S-P_k} in the $(S-P_k, k+3)$-universal set, let U_{H-P_k} denote the elements partitioned into H. If H has a r-packing P_r, there are $2r$ elements of U_{S-P_k} in P_r, denoted by $U_{P_r-P_k}$. Assume $U'_{P_r-P_k}$ denotes the elements of $U_{P_r-P_k}$ in H_1. In order to find P_r, $U'_{P_r-P_k}$ should be partitioned into H_1, and $U_{P_r-P_k} - U'_{P_r-P_k}$ should be in H_2. By lemma 9, we can construct $(U_{H-P_k}, 2r)$-universal set, whose size is bounded by $O(\log n 2^{k+3+12\log^2(k+3)+18\log(k+3)})$.

In the derandomization of algorithm RSP, the time complexity is:

$$\begin{aligned} T_r &\le \log n 2^{k+3+12\log^2(k+3)+18\log(k+3)}(T_{\lceil \frac{r}{2} \rceil} + T_{\lfloor \frac{r}{2} \rfloor}) \\ &\le \log n 2^{k+3+12\log^2(k+3)+18\log(k+3)+1} T_{\lceil \frac{r}{2} \rceil} \\ &= O((k+3)^{\log\log n} 2^{2(k+3)+4\log^3(k+3)+15\log^2(k+3)+13\log(k+3)}). \end{aligned}$$

In the practical point of view, T_r is bounded by: $O(2^{2(k+3)+4\log^3(k+3)+15\log^2(k+3)+11\log(k+3)})$.

If there exists P_{k+1}, on the basis of $(S-P_k, k+3)$-universal set and the above result, we can get the P_{k+1} deterministically with time complexity:

$$\sum_{r=0}^{\lceil \frac{k+3}{2} \rceil} \binom{3k}{r} (\log n 2^{k+3+12\log^2(k+3)+18\log(k+3)}$$
$$(2^{3k-r} + 2^{2(k+3)+4\log^3(k+3)+15\log^2(k+3)+13\log(k+3)})) = O^*(3.52^{3k}).$$ □

By lemma 1 and theorem 4, we can get the following corollary.

Corollary 1. 3-Set Packing can be solved in $O^*(3.52^{3k})$.

5 Conclusions

For the 3-Set Packing problem, we construct a $(k+1)$-packing P_{k+1} from a k-packing P_k. After further analyzing the structure of the problem, we can get the following property: for any $(0,1)$-free packing P, if the 3-Set Packing Augmentation problem can not be solved in polynomial time, each set in P should contains at least one element of P_k. On the basis of the above property, we get a randomized algorithm of time $O^*(3.52^{3k})$. Based on the derandomization method given in [10], we can get a deterministic algorithm with the same time complexity, which greatly improves the current best result $O^*(4.61^{3k})$. Our results also imply improved algorithms for various triangle packing problems in graphs [10].

References

1. Liu, Y., Lu, S., Chen, J., Sze, S.H.: Greedy localization and color-coding:improved matching and packing algorithms. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 84–95. Springer, Heidelberg (2006)
2. Downey., R., Fellows, M.: Parameterized complexity. Springer, New York (1999)
3. Jia, W., Zhang, C., Chen, J.: An efficient parameterized algorithm for m-SET PACKING. J. Algorithms 50, 106–117 (2004)
4. Koutis, I.: A faster parameterized algorithm for set packing. Information Processing Letters 94, 7–9 (2005)
5. Fellows, M., Knauer, C., Nishimura, N., Ragde, P., Rosamond, F., Stege, U., Thilikos, D., Whitesides, S.: Faster fixed-parameter tractable algorithms for matching and packing problems. In: Albers, S., Radzik, T. (eds.) ESA 2004. LNCS, vol. 3221, Springer, Heidelberg (2004)
6. Kneis, J., Moelle, D., Richter, S., Rossmanith, P.: Divide-and-Color. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 58–67. Springer, Heidelberg (2006)
7. Chen, J., Lu, S., Sze, S.-H., Zhang, F.: Improved algorithms for path, matching, and packing problems. In: Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), pp. 298–307 (2007)
8. Chen, J., Lu, S.: Improved parameterized set splitting algorithms: A probabilistic approach. Algorithmica (to appear)
9. Naor, M., Schulman, L., Srinivasan, A.: Splitters and near-optimal derandomization. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS 1995), pp. 182–190 (1995)
10. Mathieson, L., Prieto, E., Shaw, P.: Packing edge disjoint triangles: A parameterized view. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 127–137. Springer, Heidelberg (2004)